Nome: Emelly Yasmin Andrade Formigário  
Nome: Gabriélly Custódio Ferreira

Trabalho de Pesquisa Comparativa entre Padrões

MVC

No padrão MVC, os modelos não entendem visualizações ou controladores. Dessa forma, os observadores de modelo são alertados sempre que as visualizações e os controladores mudam. Então, os controladores ajudam o processo de roteamento a conectar modelos a visualizações relacionadas.

Algumas das vantagens do padrão MVC são:

* Separação de interesses (mais centralização).
* Facilita o teste e o gerenciamento de código.
* Promova o desacoplamento da camada de aplicativo.
* Melhor organização e reutilização de código.

MVP

O padrão MVP compartilha dois componentes com o MVC: o modelo e a visualização. Então, ele substitui o controlador por um apresentador. Presenter – como o nome sugere – é usado para apresentar algo. Ele permite simular visualizações com mais facilidade.

No MVP, o apresentador tem uma funcionalidade “man-in-the-middle”, pois toda a lógica de apresentação é enviada a ele. Portanto, a visualização e o apresentador no MVP também são independentes um do outro e interagem por meio de uma interface.

Os apresentadores recebem entrada do usuário por meio de visualizações. Em seguida, ele processa a ação do usuário com a ajuda do modelo, passando o resultado de volta para a visualização. O apresentador se comunica com a visualização por meio de uma interface.

MVVM

MVVM é a evolução moderna do MVC. Então, o principal objetivo do MVVM é fornecer uma separação clara entre a lógica de domínio e a camada de apresentação. Portanto, o MVVM oferece suporte à vinculação de dados bidirecional entre exibições e modelos de exibição.

O padrão MVVM permite separar a visualização e o modelo do código. Portanto, isso significa que não é necessário quando o modelo altera a visualização e vice-versa. Então, usando um modelo visual, você pode testar a unidade e testar seu comportamento lógico sem envolver a visualização.

REST

REST(Representational State Transfer, que significa Transferência Representacional de Estado) é um modelo de arquitetura e não uma linguagem ou tecnologia de programação, que fornece diretrizes para que os sistemas distribuídos se comuniquem diretamente usando os princípios e protocolos existentes da Web sem a necessidade de SOAP ou outro protocolo sofisticado.

CRUD

O acrônimo CRUD é uma abreviação de quatro operações fundamentais em sistemas de gerenciamento de banco de dados e desenvolvimento de software:

Create (Criar): Todavia, essa operação diz respeito à criação de novos registros ou entradas em um banco de dados. É o ponto de partida para inserir novos dados em um sistema.

Read (Ler): A operação de leitura envolve a recuperação de informações existentes de um banco de dados ou sistema Ademais, é usada para visualizar, acessar ou buscar dados previamente registrados.

Update (Atualizar): Essa operação permite a modificação de dados existentes. Por meio dela, é possível fazer alterações em registros já inseridos.

Delete (Excluir): A operação de exclusão envolve a remoção de registros ou dados existentes em um banco de dados. É utilizada quando se deseja eliminar informações que não são mais relevantes ou precisas.

Essas quatro operações são os pilares do CRUD e são essenciais para interagir com bancos de dados e sistemas de maneira eficiente e precisa. Elas são amplamente utilizadas em uma variedade de contextos, desde sistemas de gerenciamento de conteúdo até aplicativos de comércio eletrônico e muito mais.

SINGLETON

O padrão singleton permite que uma classe ou objeto tenha apenas uma instância e utilize uma variável global para armazenar essa instância. Você pode usar o carregamento lento (do inglês, lazy loading) para garantir que haja apenas uma instância da classe, pois ela só será criada quando você precisar.

Isso impede que múltiplas instâncias estejam ativas ao mesmo tempo, o que poderia causar bugs estranhos. Na maioria das vezes, isso é implementado no construtor. O objetivo do padrão singleton é, geralmente, regular o estado global de uma aplicação.

FACTORY METHOD

É um dos padrões de projeto mais conhecidos e tem o objetivo de ocultar detalhes sobre a criação de objetos, fornecendo uma interface para que isso aconteça. A classe que implementa a interface decidirá qual objeto será criado, dentre muitos possíveis;

OBSERVER

Tem como objetivo principal definir uma dependência um-para-muitos entre objetos. Quando um objeto muda de estado, todos os dependentes são notificados e atualizados automaticamente. Para isso, o padrão Observer mantém uma lista com seus dependentes.

STRATEGY

O padrão de projeto Strategy é um dos pilares do mundo do design de software. Ele pertence à categoria dos padrões comportamentais definidos no livro “Design Patterns: Elements of Reusable Object-Oriented Software” de Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides (também conhecido como “Gang of Four”). Ele se encaixa nessa categoria de padrões comportamentais porque se concentra em como os objetos interagem e comunicam entre si.

Em termos simples, o padrão Strategy permite que você defina uma família de algoritmos, coloque cada um deles em uma classe separada e torne esses algoritmos intercambiáveis. Isso significa que você pode alterar a estratégia usada em tempo de execução sem alterar o cliente que a utiliza.

DEPENDENCY INJECTION

A injeção de dependência é um padrão de projeto utilizado para diminuir o acoplamento entre as classes de um sistema. Em outras palavras, é uma forma de estruturar o código para que as classes dependam de abstrações, em vez de dependerem diretamente de outras classes concretas.

DIFERENÇAS E O IMPACTO DE SUA IMPLEMENTAÇÃO

Os padrões MVC, MVP e MVVM são padrões de arquitetura que definem a estrutura e a interação entre diferentes partes do software. Eles diferem principalmente na maneira como a lógica de negócios é separada da interface do usuário.

* MVC: O Controller manipula a entrada do usuário e atualiza a View e o Model conforme necessário.
* MVP: O Presenter atua como um mediador entre a View e o Model.
* MVVM: O ViewModel fornece uma abstração da View que é usada para manipular o Model.

Singleton, Factory Method, Observer, Strategy e Dependency Injection são padrões de design que definem como os objetos interagem entre si.

* Singleton: Garante que apenas uma instância de uma classe exista.
* Factory Method: Define uma interface para criar um objeto, mas deixa as subclasses decidir qual classe instanciar.
* Observer: Define uma dependência um-para-muitos entre objetos para que quando um objeto muda de estado, todos os seus dependentes são notificados e atualizados automaticamente.
* Strategy: Define uma família de algoritmos, encapsula cada um e os torna intercambiáveis. A estratégia permite que o algoritmo varie independentemente dos clientes que o utilizam.
* Dependency Injection: É uma técnica em que um objeto recebe outras dependências necessárias.

REST e CRUD são conceitos relacionados ao desenvolvimento de APIs e bancos de dados.

* REST: É um estilo arquitetônico para desenvolver serviços web.
* CRUD: Representa as quatro operações básicas (criar, ler, atualizar, excluir) que podem ser realizadas em um banco de dados.

Em termos de manutenibilidade, escalabilidade e testabilidade:

* Manutenibilidade: Todos esses padrões e conceitos, quando aplicados corretamente, podem melhorar a manutenibilidade do software ao promover a separação de responsabilidades, modularidade e reutilização de código.
* Escalabilidade: REST pode melhorar a escalabilidade ao permitir que os serviços sejam escalados independentemente. Singleton pode dificultar a escalabilidade, pois a instância única pode se tornar um gargalo.
* Testabilidade: A maioria desses padrões e conceitos melhora a testabilidade ao facilitar a escrita de testes unitários e permitir a substituição de dependências por mocks. Singleton pode dificultar a testabilidade, pois o estado global pode levar a um comportamento indeterminado.

REFERÊNCIAS

* [Padrões de projetos de software: MVC, MVP e MVVM, qual escolher? - Programadores Deprê - Programação e Tecnologia (programadoresdepre.com.br)](https://programadoresdepre.com.br/padroes-de-projetos-de-software-mvc-mvp-e-mvvm-qual-escolher/)
* [Padrões de projeto: o que são? Principais tipos e vantagens (xpeducacao.com.br)](https://blog.xpeducacao.com.br/padroes-de-projeto/)
* [REST: Conceito e fundamentos | Alura](https://www.alura.com.br/artigos/rest-conceito-e-fundamentos)
* [CRUD: Qual a sua importância no Desenvolvimento de Software? (casadodesenvolvedor.com.br)](https://blog.casadodesenvolvedor.com.br/crud-e-sua-importancia-no-desenvolvimento-de-software/)
* [Entendendo o Padrão de Projeto Strategy | by Amanda Laís | Medium](https://medium.com/@amndalsr/entendendo-o-padr%C3%A3o-de-projeto-strategy-d298ed9875de)
* [Entendendo a Injeção de Dependência: Uma Abordagem para um Código Mais Flexível e Testável - DEV Community](https://dev.to/wandealves/entendendo-a-injecao-de-dependencia-uma-abordagem-para-um-codigo-mais-flexivel-e-testavel-1ik4)